# CSCI 4131 – Internet Programming Assignment 4

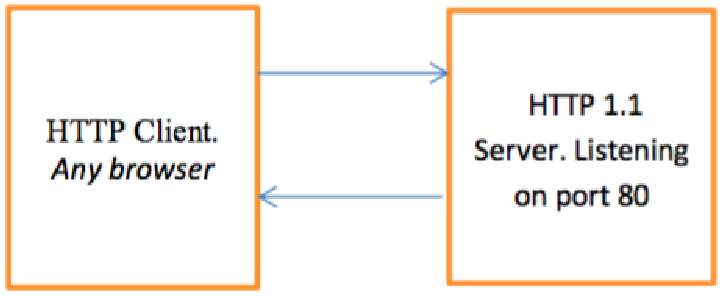
**Due 10/26/2018 at 2:00 pm (afternoon)**

*Late Submissions accepted until Saturday 10/27/2018 at 2:00am (early morning)with 15% penalty*

## 1 Description

The objective of this assignment is for you to learn the Hyper-Text Transfer Protocol (HTTP) and build a small subset of an HTTP server in Python 3. In this assignment, using Python 3 and TCP sockets, you will program some of the basic functionality of an HTTP server. You will need to go through RFC 2616 for HTTP 1.1 protocol details.

When a web client (such as Google Chrome) connects to a web server (such as ***www.google.com***) the data that is exchanged between them (e.g., HTTP messages, HTML, CSS, JavaScript, pictures, audio, video, etc.) uses the Hyper-Text Transfer Protocol.


HTTP Client. *Any browser*
HTTP 1.1 Server. Listening on port 80

The outline of how a basic HTTP server works for an HTTP GET request message is specified below.

1. An HTTP client connects to the HTTP-server and sends an HTTP request message requesting a resource to the HTTP server.
2. The request can be of type HEAD, GET, POST, HEAD, etc.
3. The HTTP Sever parses the request header fields.
4. For a GET request, the server identifies the requested resource (e.g., an HTML file) and checks if the resource exists and if it can access it. If this is the case, the server proceeds to step 5 below. Otherwise it proceeds to step 6 below.
5. The HTTP server then generates an appropriate HTTP response message. If the requested resource is found and is accessible, the HTTP server reads in the resource and builds a response message. The response includes successful (2xx) status code in the response headers along with other meta data such as the Content Type and Content Length, and includes the resource data as the message body. The sever then sends the message to the HTTP client which sent the GET request.
6. Otherwise, if the resource requested by the HTML client is not found, then the HTTP Server sends a response message with error response status code the HTML client. See section 4.4 below for a discussion about the errors your server must recognize and send a response to.

## 2 Preparation and Provided Files

The following files are provided for this assignment:

- ***403.html*** **-** this file should be sent to the client if permissions do not permit its access.
- ***404.html*** **-** this file should be sent to client if the server cannot find the requested file.
- ***private.html*** **-** this file is the private file that triggers 403 forbidden code, you can use it to test.
- ***calendar.html*** **-** replace this file with your own calendar.html file from Assignment 1.
- ***displayPix.html*** – html file containing an image to use for testing your server's capability to respond to a request for an image.
- ***psAnnouncement.html*** – html file containing an audio controls element to use for testing your servers capability to respond to a request for an audio file.
- ***Goldy_N_Announce.html*** – contains both and image and audio control element to use for testing your servers capability to respond to a request for an image and an audio file.
- ***gophers-mascot.png*** – the image file (a .png file) used by displayPix.html and Goldy_N_Announce.html
- ***flu30-081018.mp3*** – the audio file (a .mp3 file) used by psAnnouncement.html and Goldy_N_Announce.html

To give the files above proper permissions, please execute following `chmod` commands to correctly set permissions on your files after downloading them to your folder:

```
chmod 640 private.html
chmod 644 403.html
chmod 644 404.html
chmod 644 calendar.html
chmod 644 displayPix.html
chmod 644 psAnnouncement.html
chmod 644 Goldy_N_Announce.html
chmod 644 gophers-mascot.png
chmod 644 flu30-081018.mp3
```

## 3 Functionality

When you start your server, it will establish a socket and bind to a port, listening for connections. You can send a request to the server to get your calendar from your web browser by typing:

```
 <host>:<port>/mycalendar.html
```

For this assignment, when developing and testing your server, you will run the server on your local machine, so `<host>` will be `localhost` and `<port>` should default to `9001`.

**Your code should not use the `httplib` module of the Python standard library. You should do your own socket programming on this assignment. Also, you should use python 3 to develop and test your server.**

When you run your server with no arguments, your server should bind to port 9001 and serve requests. Your server should also accept one optional command line argument that specifies a port to which it should bind. Two example calls to start your server are as follows:

1. *python myServer.py*
2. *python myServer.py 9002* (server then binds to and listens on port 9002)

***Additionally, your server should log any incoming requests to STDIN***.

## 4. The HTTP Protocol

The HTTP is a protocol of non-trivial size. You will only be implementing a small, functional subset of HTTP: GET, HEAD, and POST requests. You code will also recognize and respond to a limited subset of errors discussed in section 4.4 below.

## 4.1 GET Requests

GET requests are the most commonly used HTTP requests. When your web browser requests a webpage from a server, it is issuing a GET request. Here is an example GET request that will be received by your server:

```
GET /calendar.html HTTP/1.1
Host: localhost:9001
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:33.0) Gecko/20100101
Firefox/33.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: keep-alive
```

For some request messages (HEAD and GET of just an HTML file) to be processed by your simplified server, only the first line of the request is important. Your server should either respond to the HEAD message or, for a GET of just an HTML file, extract the requested URI (calendar.html) and compose a response message that includes the requested resource and send it to the client.

For example, if you enter the following address in your browser's URL address bar:
http://localhost:9001/calendar.html
the browser should display a version of the calendar file that you developed for your first homework.

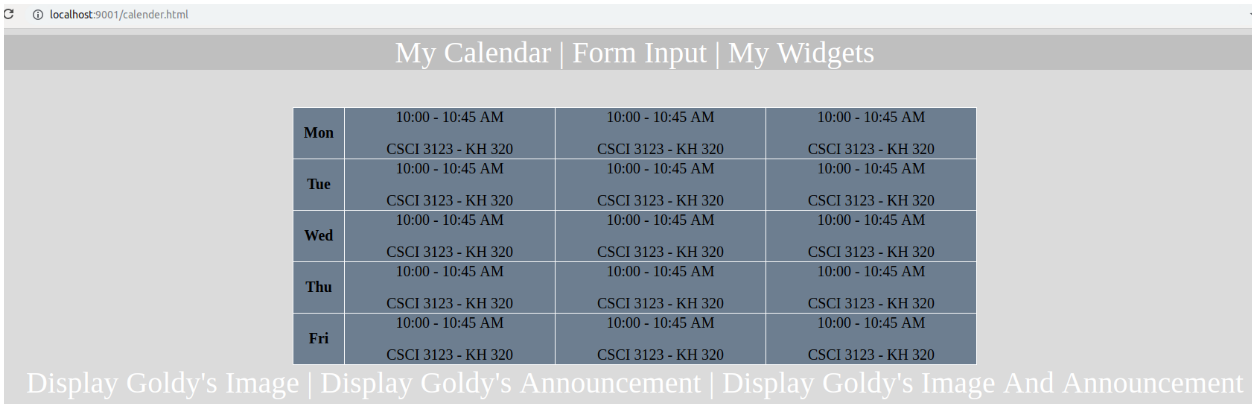
In this assignment, the resources that a client can request an html file (.html), an image file (.png) or an audio file (.mp3). When such resources are requested in the request message, your server should identify the resource type and return the resource via an http response message.

Your html file (i.e., calendar page) may contain links to external .css and .js files. You are not required to handle these two resource types. To avoid errors, you should embed your JavaScript and CSS files styling in your calendar.html file.

You will receive bonus points if your server can successfully return externally linked .css and .js files (successful means they are sent by your server and successfully obtained and applied to your calendar by the browser you are using).

As specified above, your server should be able to access the image and audio resources and return them to the browser as part of a GET request. You should add the following three links shown in the image of a calendar below to your calendar:

1) The link Display Goldy's Image should be linked to the file displayPix.html (the file contains a .png image),
2) The link Display Goldy's Announcement should be linked to the file psAnnouncement.html (the file contains .mp3 audio in an audio controls element) and,
3) The link Display Goldy's Image and Announcement should be linked to the file Goldy_N_Announce.html (the file contains a .png image and .mp3 audio). As noted above, the three html pages containing the resources are provided with this assignment.

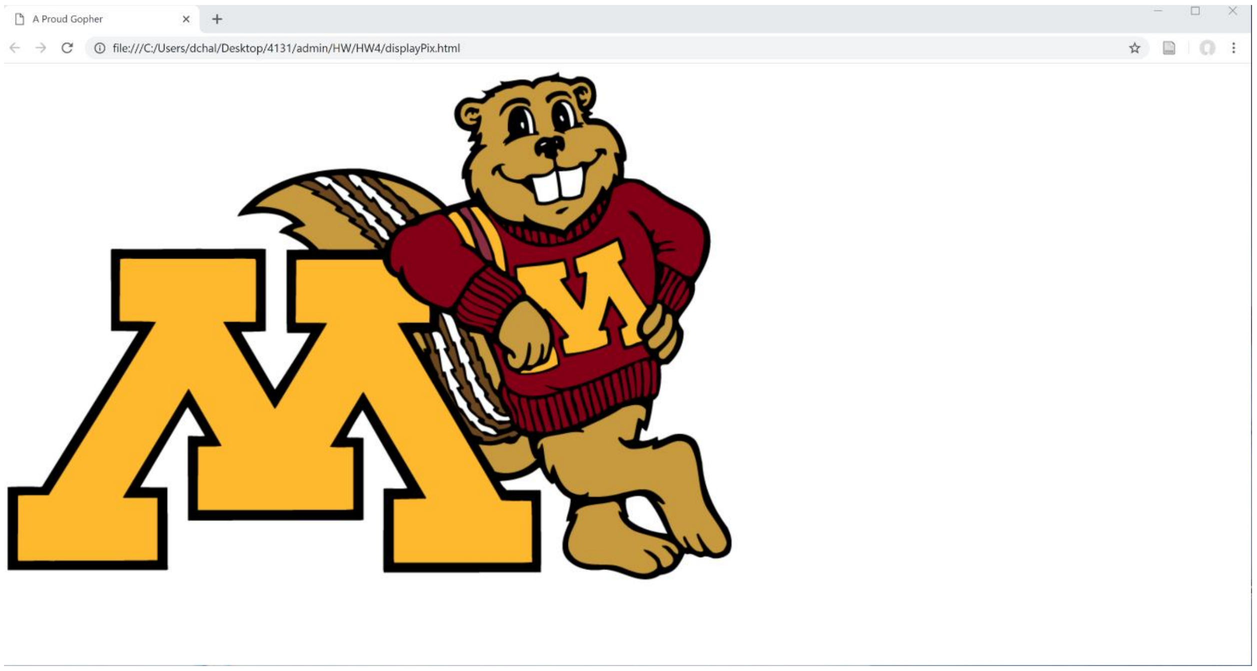

localhost:9001/calender.html

My Calendar | Form Input | My Widgets

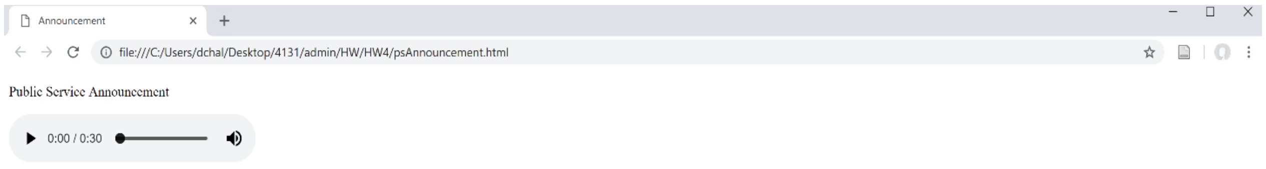
| | | | |
|---|---|---|---|
| **Mon** | 10:00 - 10:45 AM<br>CSCI 3123 - KH 320 | 10:00 - 10:45 AM<br>CSCI 3123 - KH 320 | 10:00 - 10:45 AM<br>CSCI 3123 - KH 320 |
| **Tue** | 10:00 - 10:45 AM<br>CSCI 3123 - KH 320 | 10:00 - 10:45 AM<br>CSCI 3123 - KH 320 | 10:00 - 10:45 AM<br>CSCI 3123 - KH 320 |
| **Wed** | 10:00 - 10:45 AM<br>CSCI 3123 - KH 320 | 10:00 - 10:45 AM<br>CSCI 3123 - KH 320 | 10:00 - 10:45 AM<br>CSCI 3123 - KH 320 |
| **Thu** | 10:00 - 10:45 AM<br>CSCI 3123 - KH 320 | 10:00 - 10:45 AM<br>CSCI 3123 - KH 320 | 10:00 - 10:45 AM<br>CSCI 3123 - KH 320 |
| **Fri** | 10:00 - 10:45 AM<br>CSCI 3123 - KH 320 | 10:00 - 10:45 AM<br>CSCI 3123 - KH 320 | 10:00 - 10:45 AM<br>CSCI 3123 - KH 320 |

Display Goldy's Image | Display Goldy's Announcement | Display Goldy's Image And Announcement

Pictured above: Our calendar with links to files to display the image of Goldy (`gophers-mascot.png`), display/play audio controls to play an audio file (`flu30-081018.mp3`), and display Goldy's image (`gophers-mascot.png`) and an audio control widget that should play the mp3 audio (`flu30-081018.mp3`) file about getting a "flu shot" when you select it.

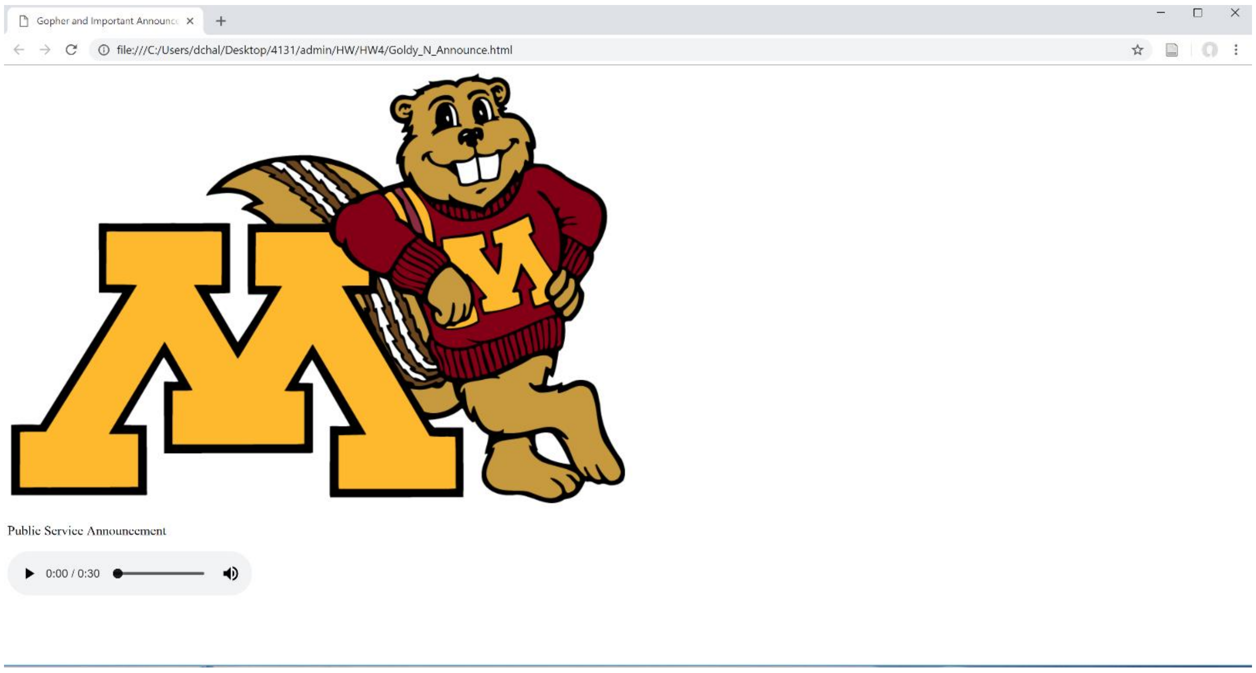

A Proud Gopher
file:///C:/Users/dchal/Desktop/4131/admin/HW/HW4/displayPix.html

**Pictured above:** The file *displayPix.html* displayed in a browser after clicking on the link Display Goldy's Image as shown in the calendar above.


Announcement
file:///C:/Users/dchal/Desktop/4131/admin/HW/HW4/psAnnouncement.html
0:00 / 0:30

Public Service Announcement

**Pictured above:** The file *psAnnouncement.html* displayed in a browser after the link on the calendar pictured above named Display Goldy's Announcement above is clicked . The audio control should play the mp3 file that is a public service announcement about getting a flu shot.


Gopher and Important Announc
file:///C:/Users/dchal/Desktop/4131/admin/HW/HW4/Goldy_N_Announce.html
Public Service Announcement
0:00 / 0:30

**Pictured above:** The file *Goldy_N_Announce.html* displayed in a browser after the link on the calendar pictured above named Display Goldy's Image And Announcement is clicked . The audio control should play the mp3 file containing a public service announcement about getting a flu shot.

## 4.2 HEAD Requests

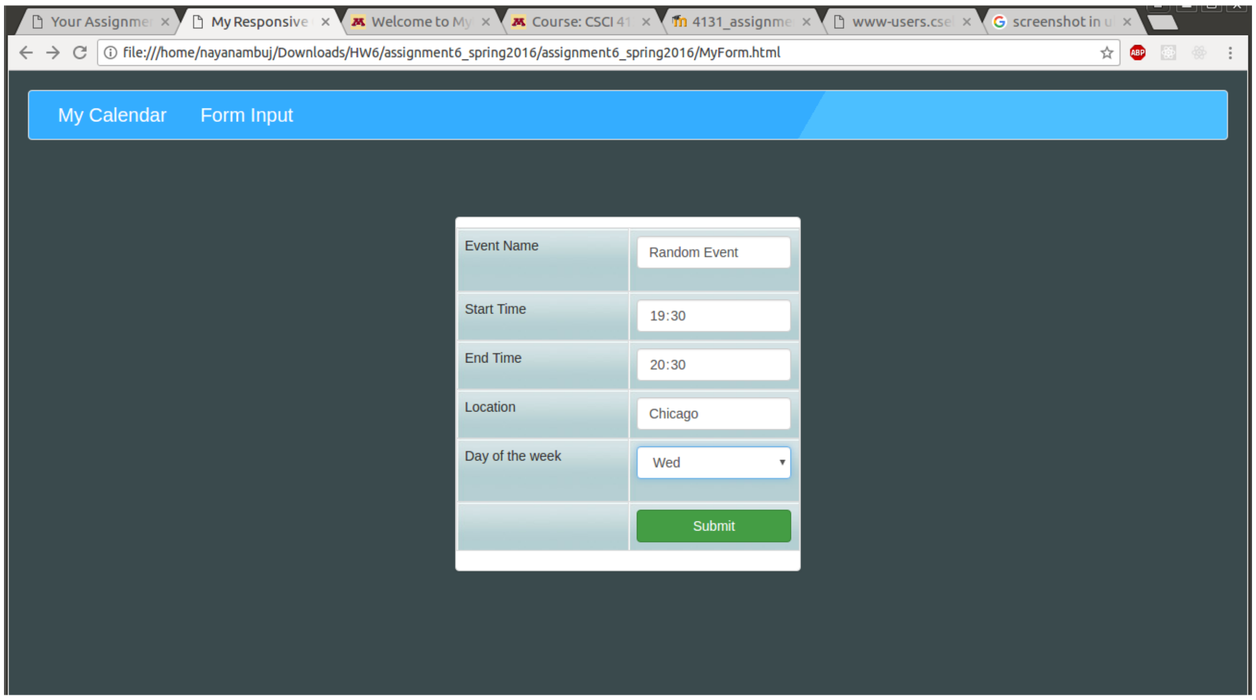
HEAD requests are almost identical to GET. Instead of returning a status code followed by the requested URI, your server should only send the status response line (for example HTTP/1.1 200 OK) and an empty message body.

The HEAD request is the easiest request to implement, but you cannot test it with your browser. You must test it with the CURL command, Telnet, or a utility to send messages such as POSTMAN.

## 4.3 POST request

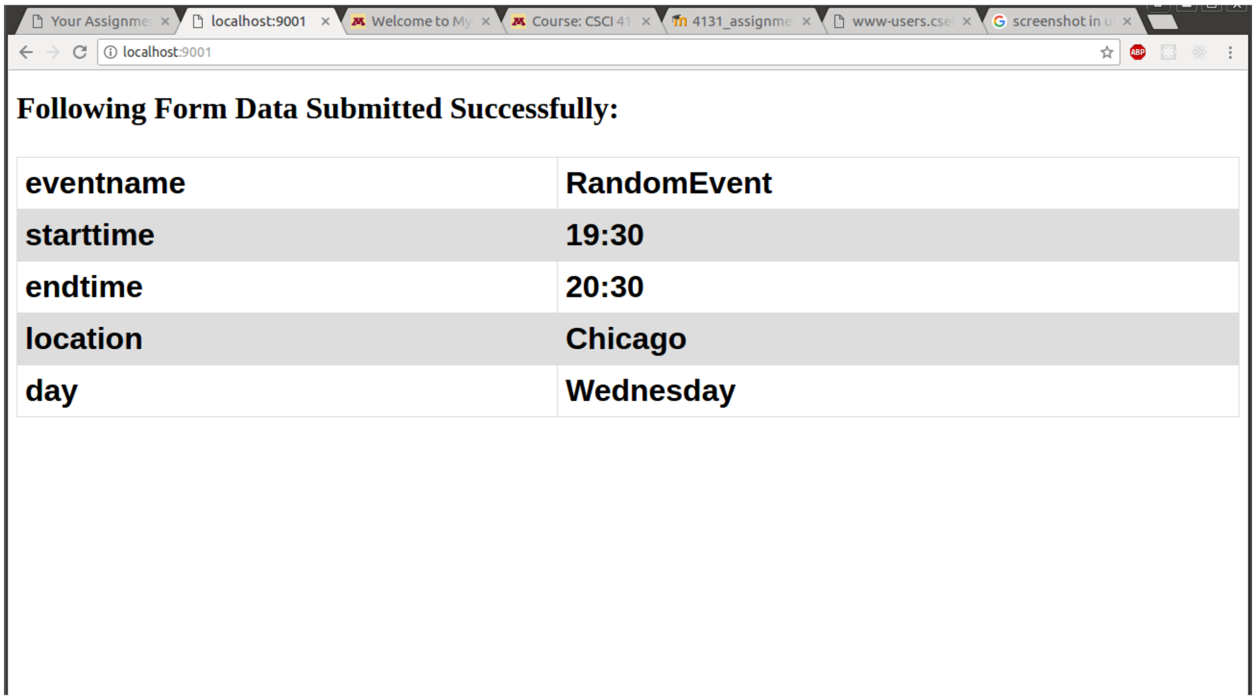
You will use the form that you developed for first homework. In the first homework, the form was submitted to https://www-users.cs.umn.edu/~shaw0162/index.php. For this homework, you will instead submit your form to your python HTTP server. You can achieve this by changing the method of the form to "post" and action as http://localhost:9001

Upon successful submission your, server should return an HTML page with all the information that was submitted on the form. Below are sample screen-shots of its expected behavior:



file:///home/nayanambuj/Downloads/HW6/assignment6_spring2016/assignment6_spring2016/MyForm.html

| | |
|---|---|
| Event Name | Random Event |
| Start Time | 19:30 |
| End Time | 20:30 |
| Location | Chicago |
| Day of the week | Wed |



localhost:9001

**Following Form Data Submitted Successfully:**

| | |
|---|---|
| eventname | RandomEvent |
| starttime | 19:30 |
| endtime | 20:30 |
| location | Chicago |
| day | Wednesday |

## 4.4 Server response to Error conditions

You will need to handle error conditions, as specified below.
**Your server should include an appropriate error message in the response message body, specific to the error condition.**
**NOTE, if we do not provide an html file specifying the error code to include in the error response message your server composes to send to the client, your server should insert an appropriate plain text error message in the response message it composes to send the client (*e.g., the client can be your Browser, Curl, Telnet, POSTMAN, etc.*).**

Your HTTP server should handle following error conditions: 403, 404, 405, and 406. It should send appropriate error responses as specified below.

1. If the requested resource is not found, your server should create a response message with a 404 error response code and send it to the requesting client.
2. If the request from the client is anything other than GET, HEAD, OR, POST, the server should compose response message containing a 405 error – method not allowed.
3. If the request from the client contains accept headers (e.g., *Accept: text/html*), and the content characteristics of the requested resource are not acceptable according to the accept headers, then the server should compose a response message with a 406 error code and send it. For example, if the accept header(s) in the request message specify only html files (*Accept: text/html* in the http request message), and the requested entity is an image file (.png), your server should compose and send an HTTP response message indicating a 406 error to the requesting client.

The GET request from the client should be constructed so it includes accept headers which accept at least three resource types: *.html*, *.png*, and *.mp3*

- For an *.html* resource, the accept header in the request message from the client should contain **text/html** or ***/***.
- For a *.png* resource, the accept header in the request message from the client should contain **image/*** or ***/***.
- For *.audio* resource, the accept header in the request message from the client should contain **audio/*** or ***/***.

## 4.6 Redirection Response

Additionally, you will also send redirection responses for certain URLs. If the request is for a resource named “umntc”, then the client should be redirected to the following location: https://twin-cities.umn.edu/. For example if the GET request is for the URL http://localhost:9001/umntc it should be redirected to the URL https://twin-cites.umn.edu . The browser should be automatically redirected to the new URL. For this, your server should send an appropriate response message with required headers. Your server should include appropriate location headers in the response. The redirection response should include the “Permanently moved” status code. Please refer to section 10.3 of RFC 2616 for details.

## 5. Testing Guidelines

To run your HTTP Server, you should pick a port number above 5000. You can test the HTTP server using a HTTP client such as a browser, curl, telnet, or POSTMAN as follows:

a) *Testing with a browser or the Linux* `Curl` *command(refer to the document CurlTesting.docx)*

i) To test your server by sending a GET request message using your browser, enter the following in your browser's address bar after starting your server so it is listening on port 9001:

http://localhost:9001/calendar.html

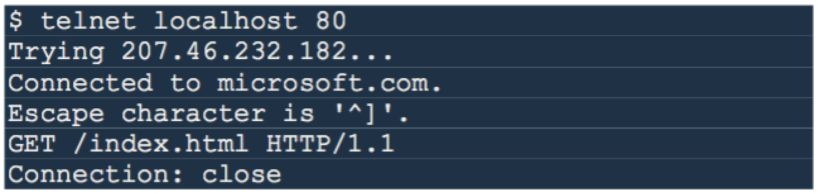
If your server composes and sends a response message correctly, your browser should display the calendar file you for the first homework (or the calendar.html file we provided with this assignment).

ii) To test your server by sending a GET request message using the Linux Curl command, type the following at your Linux command line prompt after starting your server so it is listening on port 9001:

```
curl -i -H "Accept: text/html" http://localhost:9001/calendar.html
```

The server will send its response back in a string that is displayed in your Linux terminal window.

b) *Using telnet in the Linux terminal to GET the file index.html from the directory in which your server is running (in the example shown below replace 80 with the port number your server is listening on – 9001, for example). e.g.,:*



```
$ telnet localhost 80
Trying 207.46.232.182...
Connected to microsoft.com.
Escape character is '^]'.
GET /index.html HTTP/1.1
Connection: close
```

## 6. Submission Instructions

• Submit your server program that is renamed to: `<UMN x500>myServer.py`

User *`john1234`* should submit a file named: `john1234myServer.py`

• You don't need to provide any extra files. We will use our own 403.html, 404.html, private.html, calendar.html, form.html, etc. when we test your code.

## 7. Evaluation Criteria

The HTTP server you submit will be graded out of 105 possible points on the following items:

- ➢ Server successfully establishes a socket and binds to 9001 by default. **5 points**
- ➢ Server successfully accepts (receives and uses) a parameter to assign the server to a non-default port number. **5 points**
- ➢ Server successfully accepts connections from clients and reads incoming messages. **5 points**
- ➢ Server correctly identifies GET requests, HEAD requests. **10 points**
- ➢ Server correctly responses to request for a file with an image. **5 points**
- ➢ Server correctly responds to request for a file with audio. **5 points**
- ➢ Server correctly responds to a request for a file with audio AND video. **5 points**
- ➢ Server correctly response to a request for a file with image and audio. **5 points**
- ➢ Server correctly responses with separated JavaScript and CSS. ***(5 point bonus)***
- ➢ Server correctly processes POST requests. **10 points**
- ➢ Server correctly responds with 200 and the html file (page) requested. **5 points**
- ➢ Server correctly responds with 406 and plain text message. **5 points**
- ➢ Server correctly responds with 405 and plaint text message. **5 points**
- ➢ Server correctly responds with 403 and the html file included with this assignment. **5 points**
- ➢ Server correctly responds with 404 and the html file included with this assignment. **5 points**
- ➢ Server correctly responds with a redirection request response message (301). **10 points**
- ➢ Server logs requests to STDIN. 5 **points**
- ➢ Source code is documented and readable. **5 points**
- ➢ Submission instructions are not followed correctly . (**5 point penalty – you lose 5 points)**

**Reminder: All assignments will be graded on the cselabs machines - so make sure to test your server on a cselabs machine to ensure it functions correctly as specified in the evaluation criterial above.**